PAPER *Special Section on Signal Processing*

A 158 MS/s JPEG 2000 Codec with a Bit-Plane and Pass Parallel Embedded Block Coder for Low Delay Image Transmission*

Masayuki MIYAMA[†a)], ***Member***, **Yuusuke INOIE**[†], **Takafumi KASUGA**[†], **Ryouichi INADA**[†], ***Student Members***, **Masashi NAKAO**[††], ***and*** **Yoshio MATSUDA**[†], ***Members***

SUMMARY This paper describes a 158 MS/s JPEG 2000 codec with an embedded block coder (EBC) based on bit-plane and pass-parallel architecture. The EBC contains bit-plane coders (BPCs) corresponding to each bit-plane in a code-block. The upper and the lower bit-plane coding overlap in time with a 1-stripe and 1-column gap. The bit-modeling passes in the bit-plane coding also overlap in time with the same gap. These methods increase throughput by 30 times in comparison with the conventional. In addition, the methods support not only vertically causal mode, but also regular mode, which enhances the image quality. Furthermore, speculative decoding is adopted to increase throughput. This codec LSI was designed using 0.18 μm process. The core area is 4.7×4.7 mm^2 and the frequency is 160 MHz. A system including the codec enables image transmission of PC desktop with 8 ms delay.

key words: *JPEG 2000, EBCOT, VLSI, low delay, image transmission*

1. Introduction

JPEG 2000 is an image compression standard, featuring that high compression ratio and high-quality imaging with little block noise [1], [2]. Another feature is low delay of video coding. Video coding delay of JPEG 2000 is lower than that of the methods dedicated to video compression using large buffers for rate control and bi-directional motion compensation. In addition, SNR scalability of JPEG 2000 enables to control image quality within a limited bandwidth. Thus the JPEG 2000 is used for video coding as well as still image coding.

Encoding with JPEG 2000 starts with discrete wavelet transform (DWT), followed by quantization and embedded block coding with optimized truncation (EBCOT), as shown in Fig. 1. The DWT decomposes a tile image into wavelet sub-bands. The wavelet coefficients in each sub-band are scalar quantized. After quantization, a sub-band is divided into several code-blocks. The embedded block coding includes bit-modeling and context-adaptive arithmetic coding for each code-block and forms a bit-stream. The bit-streams are truncated to minimize the overall distortion within a given target bit rate.

In the embedded block coding, a code-block is decomposed into several bit-planes. A code-block is coded from the most to the least significant bit-plane sequentially. A bit-plane coding is divisible into 3 passes. Each pass encodes or decodes all bits in the bit-plane bit by bit sequentially. Thus, the embedded block coding is originally a bit-serial, time-consuming task. A high-performance EBC architecture is necessary to develop a high-throughput JPEG 2000 codec.

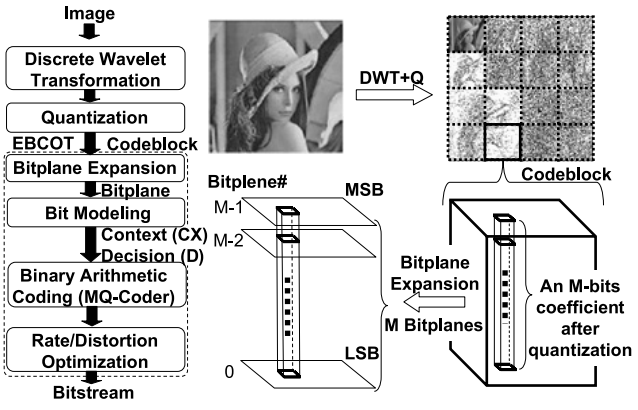


Fig. 1 JPEG 2000 overview.

The novel EBC architecture enables encoding and decoding of all bit-planes in a code-block in parallel. Three bit-modeling passes on a bit-plane coding are also executed in parallel. This architecture supports not only vertically causal mode but also regular mode, which enhances image quality. A JPEG 2000 codec LSI with an EBC based on the architecture achieves 158 M-sample/s throughput. This LSI can encode and decode not only HD (1920×1080, 30 fps, YCbCr 4:2:2, 124 M-sample/s), but also SXGA (1280×1024, 60 fps, YCbCr 4:2:2, 158 M-sample/s) video in real-time. A system including the codec enables image transmission of PC desktop with 8 ms delay.

The rest of this paper is organized as follows. Section 2 briefly introduces an EBCOT overview. Section 3 describes parallel methods of embedded block coding. Section 4 describes EBC architecture based on the parallel methods. Section 5 describes a JPEG 2000 codec LSI including the EBC. Section 6 describes a low delay image transmission system oriented to PC desktop including our JPEG 2000 codec. Section 7 concludes this paper.

Manuscript received December 10, 2007.
Manuscript revised March 14, 2008.

[†]The authors are with Kanazawa University, Kanazawa-shi, 920-1192 Japan.
[††]The author is with EIZO Nanao Co., Hakusan-shi, 924-8566 Japan.

*Part of this manuscript is first published in Proceedings of 26th Picture Coding Symposium in 2007, published by EURASIP.

a) E-mail: miyama@t.kanazawa-u.ac.jp

DOI: 10.1093/ietfec/e91–a.8.2025

2. EBCOT Overview

The EBCOT consists of an embedded block coding and a rate/distortion optimization. In the embedded block coding, quantized coefficients in each code-block are coded from the most significant bit-plane to the least significant bit-plane sequentially, as shown in Fig. 2. Bit modeling produces a pair of context and decision (CX-D) for each bit on a bit-plane. Pairs of CX-D are provided to context-adaptive arithmetic coding. A bit-stream is obtained after arithmetic coding.

Bit modeling on a bit-plane is divisible into 3 passes: a significant propagation pass (SP), a magnitude refinement pass (MR), and a cleanup pass (CU). Processing order of these passes is as above. Four adjacent rows in a bit-plane compose one stripe. Stripes in a bit-plane are scanned below from the top. Four vertically adjacent bits in a stripe composes one column. Columns in a stripe are scanned from left to right. Four bits in a column are scanned below from the top.

Each bit on the bit-plane is modeled by one of the three passes. A pass-decision for each bit depends on significance states of eight coefficients surrounding the coefficient including the bit and the state of itself. The significance state is defined for each quantized coefficient. A coefficient is significant after appearance of its first '1' in the code-block coding. A pair of CX-D corresponding to a coefficient bit also depends on the nine significant states. In the regular mode, significant states beyond a stripe boundary are referred by the pass-decision and the bit-modeling. In the vertically causal mode, significant states beyond a stripe boundary are not referred, resulting in a degradation of image quality.

Several EBCOT architectures have been proposed. One EBCOT architecture proposed in [3] includes two bit-plane coders, which encode two bit-planes in parallel. The bit-plane coder executes several bit-modeling passes also in parallel. This architecture achieves high throughput in a case that the numbers of bits processed in each pass are almost equal. But the numbers of processing bits for each pass in a bit-plane are not balanced in the usual case. This architecture requires an additional buffer, which stores pairs of CX-D for several bit-planes to increase possibility of parallel processing, between a bit-modeling and MQ coders.

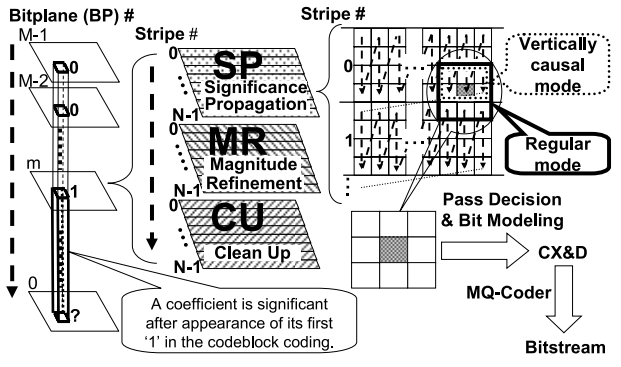


Fig. 2 Overview of embedded block coding.

Word-level parallel architecture is proposed in [4], [5]. This architecture can encode one quantized coefficient (=one word) in a step. Each bit of a quantized coefficient can be encoded in parallel because all of the neighbor significance states necessary for the current context calculation are known at encoding. However, each bit of a coefficient cannot be decoded completely in parallel at decoding. To start decoding of a bit, complete decoding of its former dependent bits is necessary. An additional mechanism to solve the neighbor dependency is required for decoding. This architecture supports only the vertically causal mode, whose neighbor dependency is simpler than that of the regular mode.

Novel parallel methods of embedded block coding and its architecture, improving throughput while supporting the regular mode, are discussed in the later sections.

3. Parallel Embedded Block Coding Methods

3.1 Formulation of Significance State

A significance state $\sigma(m,i,p)$ and a pass decision $PD(m,i,p)$ at the i-th bit position on the m-th bit-plane in a pass p are expressed by following equations:

$$\begin{aligned}\sigma(m,i,p) = {} & ((p == 1)\&(\sigma(m+1,i,3)/(NS(m,i)\&\nu(m,i)))) \\ & /((p == 2)\&(\sigma(m,i,1))) \\ & /((p == 3)\&(\sigma(m,i,1)/\nu(m,i))) \end{aligned} \quad (1)$$

$$\begin{aligned}PD(m,i,p) = {} & ((p == 1)\&(\sim \sigma(m+1,i,3)\&NS(m,i))) \\ & /((p==2)\&(\sigma(m+1,i,3))) \\ & /((p==3)\&(\sim \sigma(m+1,i,3)\& \sim NS(m,i))) \end{aligned} \quad (2)$$

$$NS(m,i) = \left(\left(\sum_{j\in N(i), j<i} \sigma(m,j,1) + \sum_{j\in N(i), j>i} \sigma(m+1,j,3) \right) \geq 1 \right). \quad (3)$$

Equation (1) represents a significance state of a coefficient at the i-th position after scanning of a pass p on the m-th bit-plane. Equation (2) represents a pass decision about a pass p at the i-th bit position on the m-th bit-plane.The pass decision $PD(m,i,p)$ becomes 1 when a bit at the i-th position on the m-th bit-plane is modeled by the pass p. Values 1, 2, and 3 of the pass p correspond to the SP, the MR, and the CU pass, respectively. Neighbor significance $NS(m,i)$ at the i-th bit position on the m-th bit-plane in Eqs. (1) and (2) is expressed by Eq. (3). The $NS(m,i)$ becomes 1 when one or above significant coefficients exist among eight neighbor coefficients of the i-th coefficient before the SP pass scanning on the m-th bit-plane. The $N(i)$ in Eq. (3) represents a set of eight neighbor position numbers of the i-th position. The number for each position is assigned according to the scan order in the bit-modeling pass. The $\nu(m,i)$ in Eq. (1) represents a quantized coefficient bit at the i-th bit position on the m-th bit-plane. The operator notations in the equations follow those of the C language. Further, context

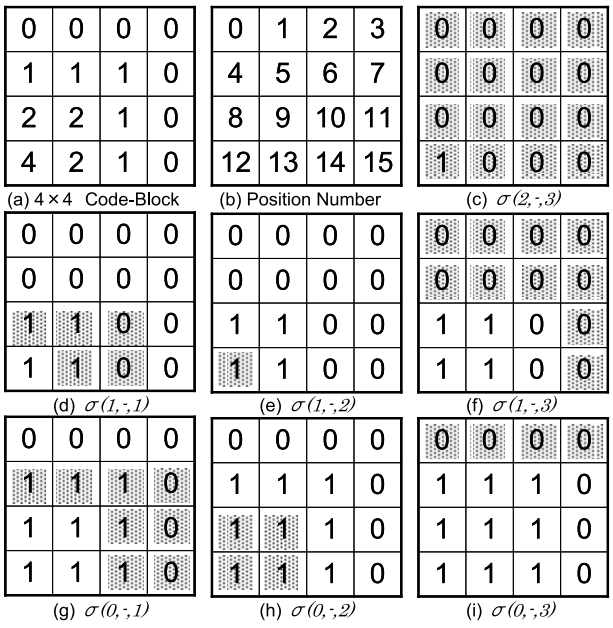


Fig. 3 Example of significance state transition.

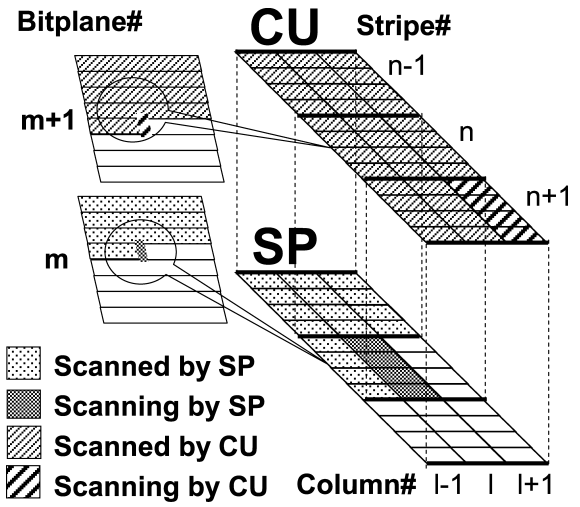


Fig. 4 Bit-plane parallel method.

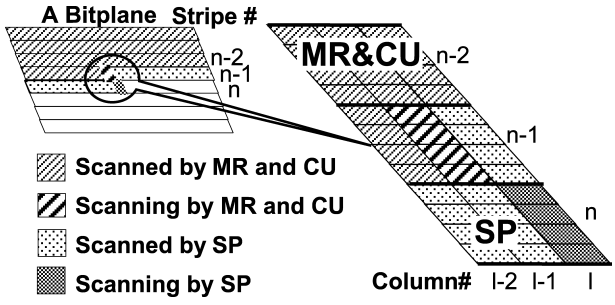


Fig. 5 Pass parallel method.

calculation of a bit in a pass requires neighbor eight significance states. The neighbor states for the former bits in the scan order must be evaluated after scanning of the current pass. The later states must be evaluated before scanning of the current pass.

Figure 3 shows an example of significance state transition about a 4×4 code-block. Quantized coefficients are shown in Fig. 3(a). Word length of the coefficients is three bits and the number of bit-planes is three. Bit position numbers and the scan order are shown in Fig. 3(b). For simplification, one row composes one stripe in this example. State transition from the CU pass on the 2nd bit-plane to the CU pass on the 0th bit-plane is shown by matrices from Fig. 3(c) through Fig. 3(i). Each matrix corresponds to a pass. An element value of the matrix represents a significance state evaluated after the pass. Shaded elements represent bits modeled by the pass. Pass decision and significance state evaluation for each bit follow Eqs. (1) and (2). Note that the most significant bit-plane is processed by the CU pass only, in general.

3.2 Parallel Methods

As shown in Fig. 4, the SP pass for l-th column in the n-th stripe on the m-th bit-plane is considered first. According to Eqs. (1) and (2), the bottom bit of the l-th column requires a significance state of a coefficient, located at the top of the $(l+1)$-th column in the $(n+1)$-th stripe and evaluated after the CU scan on the $(m+1)$-th bit-plane, for pass decision and significance state evaluation. The state is also necessary for context calculation of the bit. Thus the bit-plane parallel method overlaps the upper bit-plane coding and the lower bit-plane coding with one stripe and one column gap. With this method, all bit-planes are coded in parallel, while supporting the regular mode. The coding throughput of a code-block divisible into M bit-planes improves by M times.

As shown in Fig. 5, the MR pass and the CU pass for $(l-1)$-th column in the $(n-1)$-th stripe on the m-th bit-plane are considered next. The bottom bit of the $(l-1)$-th column requires a significance state of a coefficient, located at the top of the l-th column in the n-th stripe and evaluated after the SP scan on the m-th bit-plane, for context calculation. Thus the pass parallel method overlaps the bit-modeling passes with one stripe and one column gap.

The figure also shows the pass-decision method. The SP pass scans the lower right four bits. The MR pass and the CU pass scan the upper left four bits. The two passes can scan the same four bits simultaneously because the MR pass does not change the significance states. For the eight scanning bits shown in the figure, pass decision and significance state evaluation for each bit can be executed in a step using Eqs. (1) and (2), if all quantized coefficient bits (v) are known before processing. Then only the selected bits are modeled bit by bit. Only one cycle per one bit is required with this method. With the conventional method, by which the pass-decision for each bit is done in each pass separately, three cycles per one bit are required. This method improves throughput by three times, while supporting the regular mode.

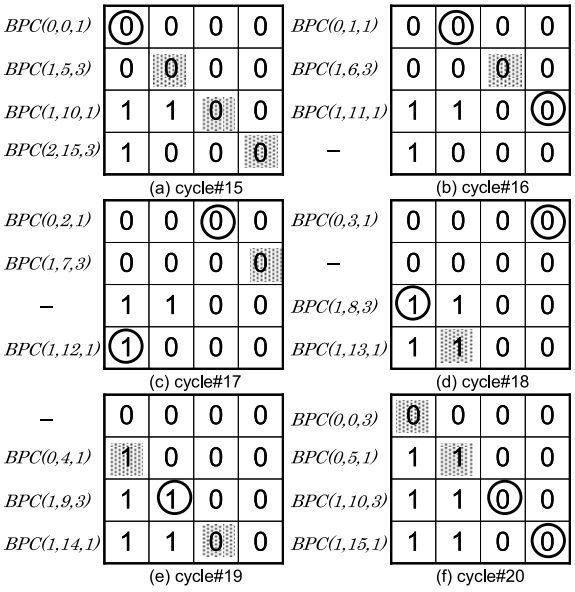


Fig. 6 Example of state transition with the proposed methods.

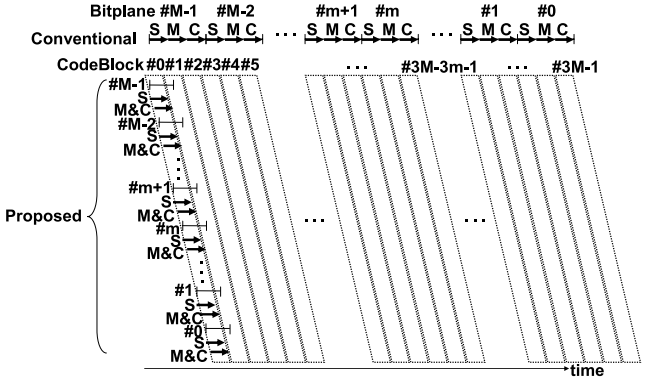


Fig. 7 Timing diagram of the proposed methods.

Figure 6 shows an example of significance state transition about the same 4×4 code-block in Fig. 3 with the proposed methods. $BPC(m, i, p)$ in the figure means that the i-th bit position on the m-th bit-plane is currently scanned by a pass p. Shaded bits are modeled by the pass. Circled bits are skipped by the pass. Bit-planes and passes are coded in parallel. Results with the proposed methods are identical to those with the sequential method. Note that significance states in the upper stripe must be referred in the lower stripe before updating them by the upper pass.

Figure 7 shows a timing diagram with these overlap methods compared to that of the conventional method. These overlap methods increase coding throughput by 3*M times and reduce the coding latency and data hold time of a code-block, resulting in smaller code-block and bit-stream buffers.

3.3 Speculative Decoding

Figure 8 shows a speculative decoding method for a bit possibly modeled by the SP pass. In a non-speculative case, pipeline bubbles occur to wait for complete decoding of the former bit modeled by the SP pass certainly. With the speculative method, decoding of a bit, which is assumed to be modeled by the SP pass, starts before complete decoding of the former bit. If the former coefficient becomes significant in the SP pass, the speculation succeeds and pipeline bubbles are eliminated. Pipeline bubbles occur with unsuccessful speculation; however, the number of bubble cycles is equal to that of the non-speculative case.

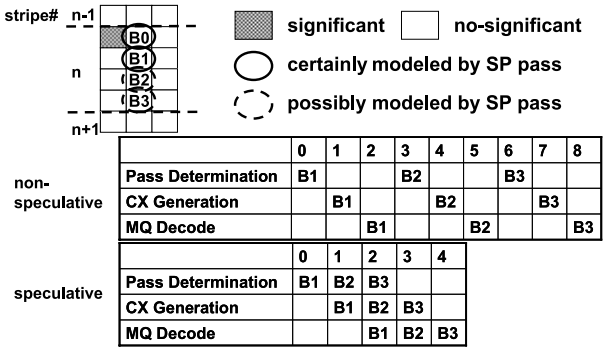


Fig. 8 Speculative decoding.

Two successive bits in the same pass are modeled using another method. Decoding of the later bit starts before complete decoding of the former bit. The method generates the later context assuming that the former bit is not significant. If the former bit becomes significant after decoding, then the context is modified in an MQ decoder. This method always eliminates bubble cycles.

4. Embedded Block Coder Architecture

4.1 Embedded Block Coder

An EBC block diagram is shown in Fig. 9. The EBC consists of a code-block buffer (CBB), bit-plane coders (BPC), and a bit-stream buffer (BSB). The CBB stores quantized coefficients by the code-block. Each bit-plane in a code-block is accessible in parallel. The number of BPCs is the same number of bit-planes in a code-block. Each bit-plane of the CBB connects two-way to a BPC one by one. The BSB stores a bit stream for each pass on a bit-plane. The BSB is divided into areas corresponding to each bit-plane.

Each BPC encodes or decodes the corresponding bit-plane at a time. At encoding, a sign bit-plane (BPS) supplies sign bits to the uppermost BPC. Each bit-plane supplies quantized coefficient bits to the corresponding BPC. A first-in-first-out buffer (FIFO) connects an upper BPC to a lower BPC. The upper BPC sends coefficient states and sign bits after complete encoding to the lower BPC via the FIFO. The coefficient state is a coding state including a significance state of the coefficient. At decoding, each BPC provides quantized coefficient bits obtained by decoding to the corresponding bit-plane. The upper BPC sends coefficient states and sign bits to the lower BPC via the FIFO. The least

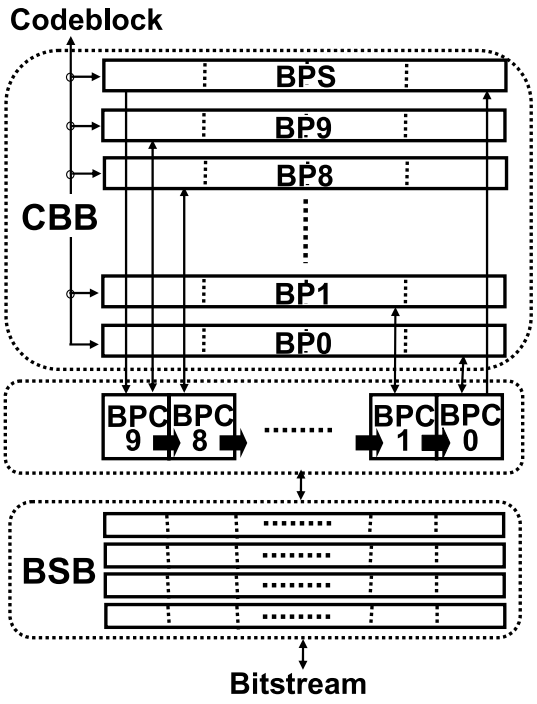


Fig. 9 Block diagram of embedded block coder.

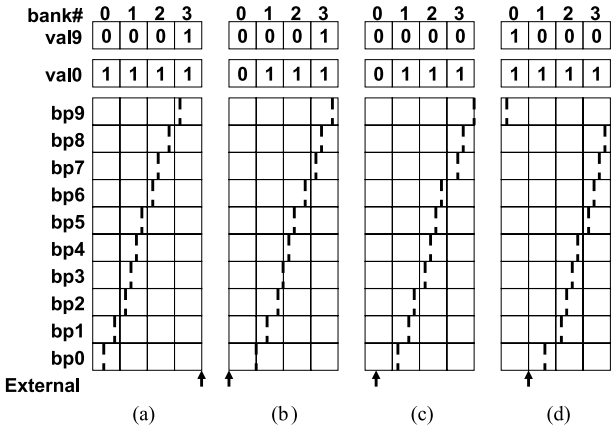


Fig. 10 Synchronization mechanism for buffer access.

significant BPC provides final sign bits to the BPS. Parallel connections between each BPC and the BPS are avoided with the FIFO.

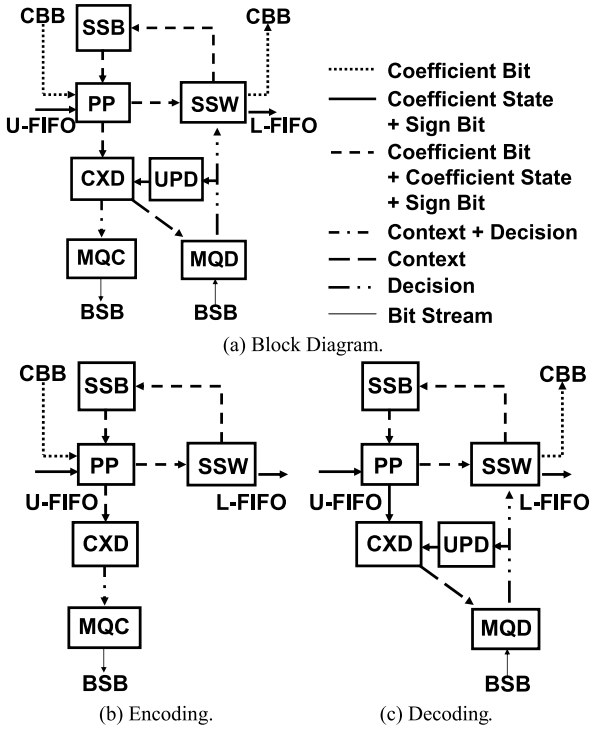
Upper bit-plane coding and lower bit-plane coding are self-synchronized by connecting BPCs with a FIFO and transferring coefficient states and sign bits from the upper BPC to the lower BPC via the FIFO. The BPC suspends coding when the upper BPC is busy and the upper FIFO is empty. The BPC also suspends coding when the lower BPC is busy and the lower FIFO is full. Each BPC executes bit-plane coding in parallel, while keeping the 1-stripe and 1-column gap. This architecture improves throughput, while supporting the regular mode.

A synchronization mechanism to access the CBB or the BSB is described here. Coding of an upper bit-plane always finishes before coding of a lower bit-plane because of the significance-state dependency. After a new code-block written to a bank from external, the most significant bit-plane in the bank can be read from internal. After writing a result corresponding to the least significant bit-plane in a bank from the internal, the code-block in the bank is legible from the external. Two state registers shown in Fig. 10 as val9 and val0, which respectively corresponds to the most significant and the least significant bit-plans, are necessary to synchronize reading from and writing to the buffer. Each bit of the register represents state of a bank. The bit value 1 means that data in the bank is valid. The register value varies according to the access to the two bit-planes from the internal and the access to a bank from the external. For example, as shown in Fig. 10(a), the bp9 starts reading from the bank3 since the corresponding val9 bit is 1. The bank0 is not accessible from the external since the corresponding val0 bit is 1. As shown in Fig. 10(b), the bp0 finishes reading from the bank0, and then the corresponding val0 bit turns to 0. The bank0 is accessible from the external since the corresponding val9 and val0 bits are both 0. The external starts writing to the bank0. As shown in Fig. 10(c), the bp9 finishes reading from the bank3, and then the corresponding val9 bit turns to 0. The bp9 can not start reading from the bank0 since the corresponding val9 bit is 0. As shown in Fig. 10(d), the external finishes writing to the bank0, and then the corresponding val9 and val0 bits turn to both 1. The bp9 start reading from the bank0 since the corresponding bit is 1. Thus only two bits are required for synchronization to access a bank that includes 10 bit-planes and a sign bit-plane.

4.2 Bit-Plane Coder

Figure 11(a) shows a BPC block diagram. The BPC consists of a PP, a CXD, an MQC, an MQD, a UPD, an SSW, and an SSB. The BPC encodes or decodes one bit-plane at a time. At first, the encoding behavior shown in Fig. 11(b) is explained. The PP is a pre-processor for bit-modeling. The PP reads coefficient states and sign bits from the SSB and the U-FIFO, which connects to the upper BPC. The PP reads quantized coefficient bits from the SSB and the CBB. The PP decides a pass for each bit, evaluates the coefficient state for each bit, and sends the neighbor coefficient states, the neighbor sign bits, and the quantized coefficient bit to the CXD. The CXD generates a pair of CX-D using the data from the PP. The MQC receives the pair of CX-D and produces a bit-stream using context-adaptive arithmetic coding. The SSW transfers the coefficient states and the sign bits from the PP to the SSB and the L-FIFO, which connects to the lower BPC. The SSW also transfers the quantized coefficient bits from the PP to the SSB. The SSB stores coefficient states, sign bits and quantized coefficient bits to use them in the next stripe coding.

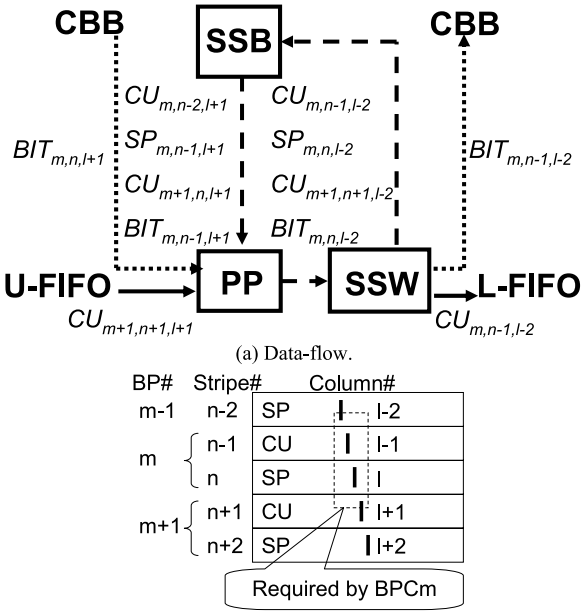
The decoding behavior shown in Fig. 11(c) is explained next. The PP behaves as the same manner at encoding, but neither reads the quantized coefficient bits from the CBB, nor evaluates the coefficient state for each bit, nor sends the



(a) Block Diagram.

(b) Encoding. (c) Decoding.

Fig. 11 Block diagram of bit-plane coder.



(a) Data-flow.

(b) Number Assignments to Bit-planes, Stripes and Columns.

Fig. 12 Data-flow diagram of bit-plane coder.

quantized coefficient bit to the CXD. The UPD evaluates and updates the coefficient states and the sign bits using decisions decoded by the MQD. The CXD generates a context using the original data from the PP and the updated data from the UPD. The MQD produces a decision using a bitstream from the BSB and the context from the CXD. At the SSW, the original coefficient states and the original sign bits provided by the PP are overwritten using the decisions provided by the MQD. The SSW writes the overwritten data to the SSB and the L-FIFO. The SSW converts the decisions into quantized coefficient bits and writes them to the CBB.

At decoding, the PP speculatively decides to model a bit by the SP pass before complete decoding of the former bit, which is modeled by the SP pass certainly. The MQD decodes the bit if the former state becomes significant after decoding. In the successful situation, the MQD sends decisions to the SSW, and the SSW writes the overwritten data to the SSB. If the former state does not become significant after decoding, the MQD does not decode the bit. In the unsuccessful situation, the MQD informs the SSW that the bit is not modeled by the SP pass, and the SSW writes the original data to the SSB.

When two successive bits in the same pass are decoded, the CXD generates the later context assuming the former bit is not significant before complete decoding of the former bit. The CXD informs the MQD that the later bit is modeled with the assumption. The MQD modifies the later context if the former state becomes significant. This method simplifies the MQD logic comparing with that of the method generating the true context again from the neighbor significance states. Re-calculation of the context and the MQ decoding can be included in the same pipeline stage; thereby, the decode pipeline dose not stall.

Figure 12 shows the BPC data flow in detail. The upper BPC sends four coefficient states and four sign bits located in the $(l+1)$-th column of the $(n+1)$-th stripe on the $(m+1)$-th bit-plane after the CU scan (denoted as $CU_{m+1,n+1,l+1}$) via the U-FIFO to the PP. The CBB sends four quantized coefficient bits located in the $(l+1)$-th column of the n-th stripe on the m-th bit-plane (denoted as $BIT_{m,n,l+1}$) to the PP. The SSB sends $CU_{m+1,n,l+1}$, $SP_{m,n-1,l+1}$, $CU_{m,n-2,l+1}$ and $BIT_{m,n-1,l+1}$ to the PP. The PP executes the SP pass for four bits located in the l-th column of n-th stripe on the m-th bit-plane using $CU_{m+1,n+1,l-1\sim l+1}$, $CU_{m+1,n,l-1\sim l+1}$, $SP_{m,n-1,l-1\sim l+1}$ and $BIT_{m,n,l}$. At the same time, the PP executes the MR pass and the CU pass for four bits located in the $(l-1)$-th column of the $(n-1)$-th stripe on the m-th bit-plane using $SP_{m,n,l-2\sim l}$, $SP_{m,n-1,l-2\sim l}$, $CU_{m,n-2,l-2\sim l}$ and $BIT_{m,n-1,l-1}$. According to Eq. (2), the MR pass and the CU pass require the significance states evaluated before the SP scan for the pass decisions. The MR pass and the CU pass can decide each pass using the significance states evaluated after the SP scan, by transforming Eq. (2) as follows:

$$PD1(m,i) = PD(m,i,1) = \sim\sigma(m+1,i,3)\&NS(m,i) \quad (4)$$

$$\begin{aligned} PD2(m,i) &= PD(m,i,2) = \sigma(m+1,i,3) \\ &= \sim PD1(m,i)\&\sigma(m,i,1) \end{aligned} \quad (5)$$

$$\begin{aligned} PD3(m,i) &= PD(m,i,3) = \sigma(m+1,i,3)\& \sim NS(m,i) \\ &= \sim PD1(m,i)\& \sim \sigma(m,i,1). \end{aligned} \quad (6)$$

The PP sends $CU_{m+1,n+1,l-2}$, $SP_{m,n,l-2}$, $CU_{m,n-1,l-2}$ and $BIT_{m,n,l-2}$ to the SSW. The SSW sends $CU_{m,n-1,l-2}$ to the lower BPC via the L-FIFO to use them on the $(m-1)$-th bit-plane coding. The SSW sends $CU_{m+1,n+1,l-2}$, $SP_{m,n,l-2}$,

$CU_{m,n-1,l-2}$ and $BIT_{m,n,l-2}$ to the SSB to use them in the SP pass for the $(n+1)$-th stripe, the MR pass for the n-th stripe, and the CU pass for the n-th stripe. The data over four stripes indicated in Fig. 12(b), which are necessary for the pass parallel processing on the m-th bit-plane, are provided by the mechanism as above. The proposed BPC architecture executes the three passes in parallel and improves throughput by three times, while supporting the regular mode.

With a conventional method, three bits represents a coefficient state, which is composed of a significance bit, a visited once bit, and an MR coded bit [6]. With our method, the coefficient state is represented by two bits instead of the three bits. State 0 represents that the coefficient is not significant. State 1 represents that the coefficient is significant and the corresponding bit has not been scanned by the MR pass in the same bit-plane on which the coefficient becomes significant in the SP pass. State 2 represents that the coefficient is significant and the corresponding bit has not been modeled by the first MR pass. State 3 represents that the coefficient is significant and the corresponding bit has been modeled by the first MR pass. The $PD1$ in Eqs. (5) and (6), and the first MR pass completion can be evaluated with this method. Moreover, the SSB stores coefficient states, sign bits, and quantized coefficient bits for only three stripes, not for all stripes, on a bit-plane. These methods reduce the SSB capacity.

5. JPEG 2000 Codec LSI

Figure 13 shows a JPEG 2000 codec block diagram. The circuit contains a VIOIF, an ITRAN, an EBC, and a CPUIF. The ITRAN contains a CCNV, an LB, a DWT, a WB, and a Q. The CCNV is an RGB–YCbCr color converter. The DWT executes DWT or inverse DWT. The Q executes quantization or inverse quantization. These circuits are reconfigurable and can operate in both directions, resulting in gate reduction. The EBC executes embedded block coding and distortion calculation. In addition to the JPEG 2000 rate-distortion optimization within a tile, an external controller progressively enhances image quality of a whole frame beyond tile boundaries within a limited bandwidth. The JPEG 2000 codec achieves throughput of one sample per clock cycle.

The JPEG 2000 codec LSI was designed using 0.18 μm process. Table 1 shows the LSI specification. Figure 14 shows a plot diagram of the codec LSI. The estimated operating frequency and power consumption of the codec are 160 MHz and 1.5 W, respectively.

6. Image Transmission System

Video compression technique for image transmission such as MPEG-2 or H.264 achieves high compression ratio. However its delay is large due to large buffers for rate control and bi-directional motion compensation. Large delay is a disadvantage in the case of interactive image transmission such as PC desktop. We propose a novel image transmission

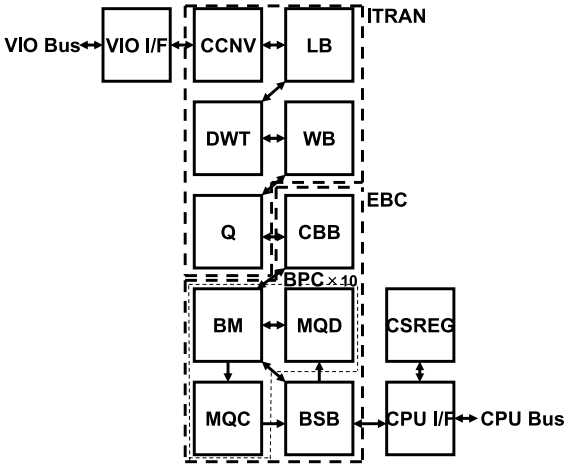


Fig. 13 Block diagram of JPEG 2000 codec.

Table 1 Specification of JPEG 2000 codec.

Function	JPEG2000 codec, regular mode supported
Performance	158 M-Sample/s (SXGA 1280 x 1024, 60 fps, YCbCr4:2:2)
Tile size	128 x 128 pixels
Codeblock size	32 x 32 pixels
Gate count	1,084 k
SRAM	625 kb (LB(S): 32 kb, WB(S): 426 kb, CBB(D): 45 kb, SSB(D): 13 kb, BSB(S): 82 kb, (S): Single Port, (D): Double Port)
Area (Core)	4.7 x 4.7 mm^2
Frequency	160 MHz
Process	0.18 um, 6 metal layers

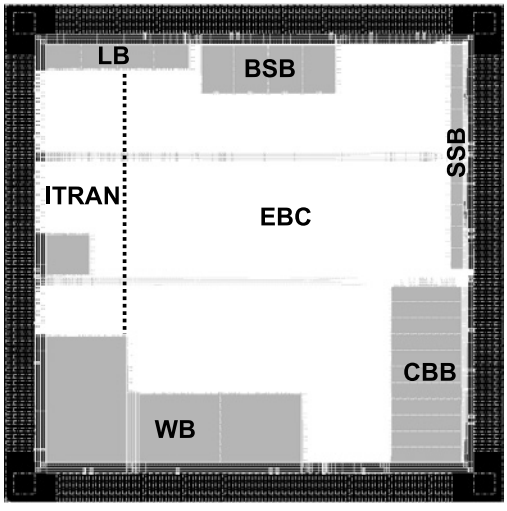


Fig. 14 Plot diagram of JPEG 2000 codec.

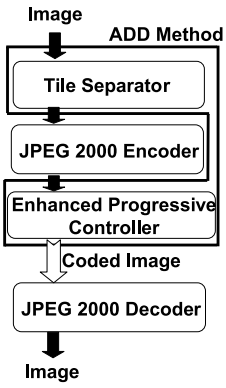
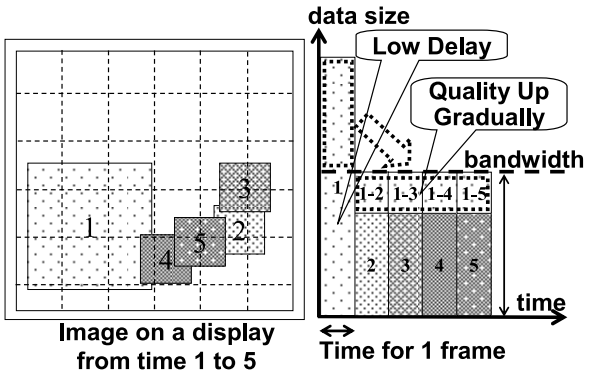
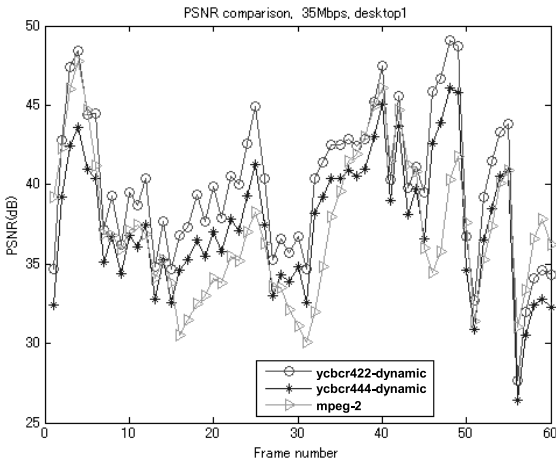


Fig. 15 Proposed method of image transmission.

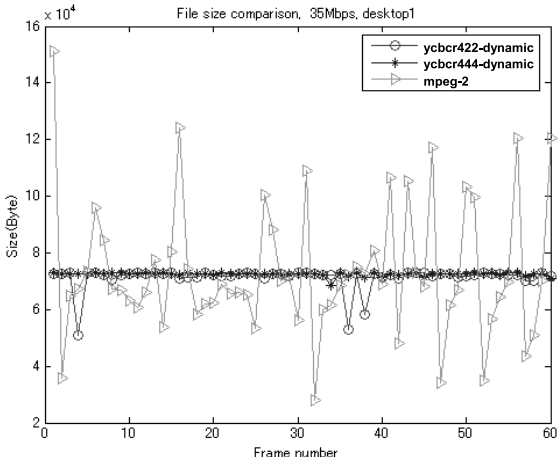


(a) Change of Image on a PC Display. (b) Transmission Data Size.

Fig. 16 Example of image transmission with proposed method.



(a) PSNR Comparison.



(b) File Size Comparison

Fig. 17 Simulation results.

method characterized by low delay and high quality within a limited bandwidth [7]. Figure 15 shows an image transmission method with ADD method and JPEG 2000. The ADD method consists of a tile separation and an enhanced progressive control. The tile separation divides a frame into tiles and finds tiles updated from the previous frame. Only the updated tiles are encoded with the JPEG 2000 encoder. The enhanced progressive control generates JPEG 2000 packets from bit-streams provided by the JPEG 2000 encoder. The controller sorts packets in the order according to the contribution toward the image quality of a whole frame beyond tile boundaries. Independent application of JPEG 2000 R/D optimization to the image quality control of a whole flame is not sufficient because its optimization is within a tile. Then packets are sent to the receiver in the determined order. The receiver decodes coded data with JPEG 2000 decoder and displays the frame.

Figure 16 shows an example of image transmission oriented to PC desktop. Figure 16(a) shows change of image on a PC display from time 1 to time 5. Figure 16(b) represents relation between transmitted data size and time. At time 1, image changes extensively and the data size is large beyond the bandwidth. All packets generated at time 1 cannot be transmitted in time 1 within a limited bandwidth. The enhanced progressive controller determines transmission order of the packets. The packets with higher contribution to a frame quality are transmitted at time 1 immediately. The packets with lower contribution are transmitted from time 2 to time 5 using the remaining space of the bandwidth, resulting in upgrade of image quality progressively. The proposed method achieves both low delay and high quality of image.

We examined the proposed method comparing with MPEG-2 by simulation. A sequence of PC desktop images (SXGA 1280×1024, 60 fps, YCbCr 4:2:2) including a VGA 30 fps video window overlaid on a web browser window was simulated as a test sequence. Note that the sequence is intended for typical PC desktop, not a full video of the SXGA 60 fps resolution. Figure 17(a) represents a relation between PSNR and the frame number. The PSNR of our method (ycbcr4:2:2-dynamic) exceeds that of MPEG-2 (M = 1, N = 15, YCbCr4:2:0, 35 Mbps) over almost all the frames. Figure 17(b) represents a relation between file size and the frame number. The file size of MPEG-2 sometimes exceeds the bandwidth (35 Mbps), while that of our method is always within a bandwidth. Those MPEG-2 frames exceeding a bandwidth are I-frames, resulting in large delay of transmission. Low delay and high quality of our method

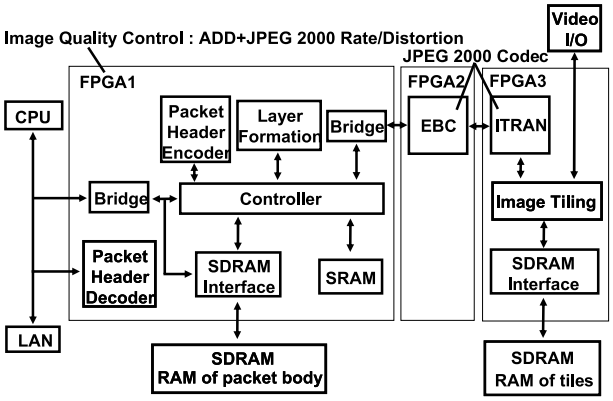


Fig. 18 Block diagram of prototype board.

was confirmed with this simulation.

A relation between our image transmission method and our JPEG 2000 codec is discussed here. A tile size is discussed at first. In the ADD method, as the tile size is smaller, the block noise grows larger. As the tile size is larger, the data size of transmission grows larger. (Consider only a mouse cursor moving on a PC display.) The adequate size of a tile obtained by the simulation is 128 pixels by 128 pixels. We determine the tile size of the codec in this way. The way of chip partition is discussed secondly. The enhanced progressive control is an expansion of the JPEG 2000 R/D optimization. Independent implementation of these controllers is inefficient. We assign distortion calculation for each coding pass to the JPEG 2000 codec. We assign the JPEG 2000 R/D optimization, which use the distortion to form packets, to the enhanced progressive controller. The third is a coding performance of the codec. The proposed method is oriented to image transmission of PC desktop. Our codec can handle SXGA 60 fps video, which is typical resolution for PC desktop, in real-time. In addition, low latency of the EBC architecture contributes to that of the system.

A prototype board of the image transmission system with the ADD method and JPEG 2000 was developed using FPGAs [8]. Figure 18 depicts the block diagram of the board. This hardware consists of three FPGAs. The JPEG 2000 codec is implemented on the FPGA2 and the FPGA3. The enhanced progressive controller is implemented on the FPGA1. We did an experiment of the image transmission system for PC desktop connecting the two boards together, and 8 ms delay from input to output of the system was obtained with the experiment. One frame corresponds to 16.7 ms at 60 fps, thus the 8 ms delay is short enough for the application.

7. Conclusion

This paper presented a JPEG 2000 codec LSI with an embedded block coder based on a bit-plane and pass-parallel architecture. An upper bit-plane and a lower bit-plane coding overlap in time with 1-stripe and 1-column gap. Bit-modeling passes also overlap in time with the same gap. These methods support not only the vertically causal mode but also the regular mode to enhance image quality, while providing short latency and high throughput of embedded block coding. Furthermore, speculative decoding is used to increase the throughput. The codec was implemented with 0.18 μm process. The core area is 4.7×4.7 mm^2 and the estimated frequency is 160 MHz. The codec achieves 158 MS/s throughput and was applied to image transmission system oriented to PC desktop with both low delay and high image quality. Quality improvement of the system for full video is a future work.

Acknowledgments

This work is supported by VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Cadence Design Systems, Inc. and Synopsys, Inc.

References

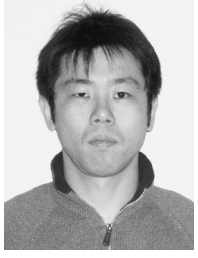
[1] "Information technology; JPEG 2000 image coding system — Part 1: Core coding system," ISO/IEC 15444-1, Aug. 2000.
[2] T. Acharya and P.-S. Tsai, JPEG2000 Standard for Image Compression, Wiley-Interscience, 2004.
[3] H. Yamauchi, S. Okada, K. Taketa, and T. Ohyama, "A single-chip JPEG 2000 encode processor capable of compressing D1-Images at 30 frame/s without tile division," IEICE Trans. Electron., vol.E87-C, no.4, pp.448–456, April 2004.
[4] H.-C. Fang, Y.-W. Chang, T.-C. Wang, C.-J. Lian, and L.-G. Chen, "Parallel embedded block coding architecture for JPEG 2000," IEEE Trans. Circuits Syst. Video Technol., vol.15, no.9, pp.1086–1097, Sept. 2005.
[5] Y.-W. Chang, C.-C. Cheng, C.-C. Chen, H.-C. Fang, and L.-G. Chen, "124 MSamples/s pixel-pipelined motion-JPEG 2000 codec without tile memory," IEEE Trans. Circuits Syst. Video Technol., vol.17, no.4, pp.398–406, April 2007.
[6] K. Andra, C. Chakrabarti, and T. Acharya, "A high-performance JPEG2000 architecture," IEEE Trans. Circuits Syst. Video Technol., vol.13, no.3, pp.209–218, March 2003.
[7] M. Nakao, M. Kita, and M. Miyama, "High quality image compression system for low delay and real-time wireless transmission," SID2006, June 2006.
[8] M. Nakao, M. Kita, and T. Matsui, "Low delay and real-time image transmission hardware for remote desktop," SID2007, June 2007.

Masayuki Miyama was born on March 26, 1966. He received a B.S. degree in Computer Science from the University of Tsukuba in 1988. He joined PFU Ltd. in 1988. He received an M.S. degree in Computer Science from the Japan Advanced Institute of Science and Technology in 1995. He joined Innotech Co. in 1996. He received a Ph.D. degree in electrical engineering and computer science from Kanazawa University in 2004. He is an assistant professor at Kanazawa University Graduate School of Science and Technology. His present research focus is VLSI designs for real-time image processing.

Yuusuke Inoie was born in Ishikawa, Japan, in 1983. In 2006, he received the B.S. degree in Department of Electrical and Electronic Engineering from Kanazawa University, where he is currently working toward the M.S. degree in Division of Electrical and Computer Engineering. His research interests include algorithms and architecture for embedded block coding with optimized truncation (EBCOT), and related VLSI designs.

Takafumi Kasuga was born in Nagano, Japan, in 1983. In 2006, he received the B.S. degree in Department of Electrical and Electronic Engineering from Kanazawa University, where he is currently working toward the M.S. degree in Division of Electrical and Computer Engineering. His research interests include algorithms and architecture for discrete wavelet transform (DWT), and related VLSI designs.

Ryouichi Inada was born in Fukui, Japan, in 1984. In 2007, he received the B.S. degree in Department of Electrical and Electronic Engineering from Kanazawa University, where he is currently working toward the M.S. degree in Division of Electrical and Computer Engineering. His research interests include algorithms and architecture for embedded block coding with optimized truncation (EBCOT), and related VLSI designs.



Masashi Nakao received the master degree in Physical Electronics from Tokyo Institute of Technology in 2002. After then, he is with Eizo Nanao Corporation to research and develop image processing and digital integrated circuit.

Yoshio Matsuda was born in Ehime, Japan, on October 26, 1954. He received the B.S. degree in physics and the M.S. and Ph.D. degrees in applied physics from Osaka University in 1977, 1979, and 1983, respectively. He joined the LSI Laboratory, Mitsubishi Electric Corporation, Itami, Japan, in 1985. He was engaged in development of DRAM, advance CMOS logic, and high frequency devices and circuits of compound semiconductors. Since 2005, he has been a professor of Graduate School of Natural Science and Technology at Kanazawa University, Japan. His research is in the fields of integrated circuits design where his interests have includes multimedia systems, low power SoCs, and image compression processors.